# Group D Milestone 2: Technical Analysis & Flask Prototype for TriniCarsForSale

**INFO3608 E-Commerce**

March 26, 2025

## 1. Introduction

TriniCarsForSale is a local online platform in Trinidad and Tobago that connects vehicle buyers and sellers. Building on Milestone 1 (which covered the business model, target customers, competitors, and initial e-commerce viability), this Milestone 2 document focuses on the **technical roadmap** for modernizing the platform and delivering a **Flask-based prototype**. The goal is to improve usability, trust and scalability while keeping the project scope realistic for a university assignment.

## 2. Internet Assessment

### 2.1 Current Modes of Access

- **Broadband/Fiber** for home users
- **Mobile Networks (3G/4G/5G)** for on-the-go access
- **Public Wi-Fi** in urban centers

### 2.2 Benefits

- **Broadband:** Reliable, supports images/videos for vehicle listings
- **Public Wi-Fi:** Expands reach to those without personal internet

### 2.3 Challenges & Limitations

- Variable speeds in some areas
- Security vulnerabilities on public networks

### 2.4 Implications for TriniCarsForSale

- Prioritize **fast loading times** and **efficient image handling**
- Implement basic **encryption** (HTTPS) to protect user data

## 3. Software Requirements

### 3.1 Business Objectives

- Provide a **modern, user-friendly** platform to list and browse vehicles
- Increase **buyer-seller trust** with basic verification measures
- Enable **premium listing** and **banner advertising** as revenue streams
- Collect basic analytics (e.g., most viewed listings) for strategic decisions

### 3.2 System Functionalities

- **User Management:** Registration, login, profile management
- **Listing Management:** Create/edit/delete vehicle listings with images and details
- **Search & Filtering:** Filter by make, model, price, year, etc.
- **Payment Integration:** Simple checkout for premium listing fees
- **Administrative Dashboard:** Manage user accounts and moderate suspicious listings

### 3.3 Information Requirements

- **Vehicle Data:** Make, model, price, mileage, images
- **User Data:** Name, email, password (hashed), phone number
- **Payment Data:** Transaction details for premium listings
- **Analytics:** Page views, listing popularity, user metrics

### 3.4 System Design Specifications

- **Technology Stack:**
  - Backend: Flask (Python)
  - Frontend: Jinja
  - Database: SQLite for simplicity (can upgrade later)
- **Hosting & Deployment:**
  - Local development for testing
  - Optional cloud deployment (Render)

## 4. Prototype Overview

### 4.1 Prototype Goals

- Demonstrate **core user flows**: viewing car listings, creating an account, posting a listing
- Present a **responsive interface**

### 4.2 Tools & Implementation

- Flask for routing and server logic
- Jinja templates for dynamic HTML
- Bootstrap for responsive styling
- SQLite for data storage

### 4.3 Sample Routes

- `/` – Home page (featured or premium listings)
- `/login` (GET & POST) – User login
- `/register` (GET & POST) – User registration
- `/listings` – List all vehicles, with search/filter
- `/listings/new` – Create a new vehicle listing

- `/listings/<id>` – View details of a specific listing
- `/admin` – Admin panel (optional)

# 5. Simplified Testing Strategy

## 5.1 Unit Testing (Minimal)

- Use Python's `unittest` or `pytest` to test core Flask routes
- Example: check `/login` returns a 200 status, verify registration logic

## 5.2 Manual Functional Testing

- Test each user flow: register, login, create listing, logout
- Ensure images upload correctly
- Check error handling (e.g., invalid form input)

## 5.3 Basic Acceptance Testing

- Ask a small group of classmates to use the site briefly
- Collect feedback on usability, layout, and bugs

## 5.4 Smoke Testing

- After updates, quickly confirm main routes (home, listings, login) still work

## 5.5 Simple Security Checks (Optional)

- Ensure passwords are hashed
- Test a few invalid inputs (e.g., `<script>`) to confirm safe handling

# 6. Delivery Strategy

## 6.1 Phased Rollout (University Demo)

- Deploy a test version to Render
- Share the link for feedback

### 6.2 User Awareness (Hypothetical)

- Social media announcements (Facebook, Instagram)
- Partnerships with local auto shops or insurance companies

### 6.3 Maintenance & Updates

- Use version control (GitHub) for updates
- Document known issues in a README

## 7. Payment Systems (Simple Approach)

### 7.1 Mock Integration

- Demonstrate with mock payment route
- Show how users can "pay" for premium listings without real transactions

### 7.2 Local Support & Challenges

- Real-world: integrate local providers (WiPay, Linx, Credit and Debit cards)
- For assignment website: a "fake" success page

### 7.3 Security Measures

- Use HTTPS in production
- Validate payment details on the server side

## 8. Growth Strategy

### 8.1 Technical Scalability

- Migrate from SQLite to MySQL/PostgreSQL if user base grows
- Deploy on AWS or Azure with load balancing for high traffic

### 8.2 Feature Expansion

- Add user reviews/ratings for sellers
- Advanced search by color, engine type, etc.
- Vehicle history integration (if data becomes available)

## 9. Security Considerations (Basic Level)

### 9.1 Data Security

- Hash passwords with `generate_password_hash`
- Validate form inputs to avoid malicious data

### 9.2 Threat Levels

- **High Risk:** Payment transactions (use reputable gateways)
- **Medium Risk:** User data leaks (SSL/TLS, limited data retention)
- **Low Risk:** Public listings (basic moderation for scams)

### 9.3 Local & International Standpoints

- **Local:** Comply with T&T data protection guidelines
- **International:** Basic GDPR awareness if dealing with EU users